**Social and Technological Networks** Edinburgh, 2018

# Notes 2. Cascades, approximations and submodular optimizations.

*Rik Sarkar* *Notes*

**Optimizations.** In class we discussed optimization problems and approximations. For a given function $f(\cdot)$, a *maximization* problem is to find the input $x$ to $f$ that achieves the maximum possible $f(x)$. Similarly, a *minimization* problems about finding the input to minimize $f$.

For the influence maximization problem, the input $x$ too the form of subsets of $V$. This creates a computational challenge, since there are many possible subsets of $V$. We were looking for subsets of size $k$, but even then, there are $\binom{n}{k}$ possibilities, which is can be very large for a large $k$.

**Q 1.** *Suppose $k = n/2$. Show that the number of possible subsets of size $k$ is at least $2^{\Omega(n)}$.*

**Approximations.** For a maximization problem, if $f$ achieves its maximum value for input $x^*$ and $f(x^*) = OPT$, then a $c$-approximation algorithm finds an $x$ such that $f(x) \geq c \cdot OPT$. The value for $c$ in this case will be a positive fraction less than $1$.

For a minimization problem, if $f$ achieves its minimum value for input $x^*$ and $f(x^*) = OPT$, then a $c$-approximation algorithm finds an $x$ such that $f(x) \leq c \cdot OPT$. The value of $c$ will be greater than $1$.

In both cases, $c$ is called the approximation factor.

**Q 2.** *Suppose we want to find shortest paths. Is this is a maximization or minimization problem? What can you say about the approximation factor of an approximation algorithm for this problem?*

**Q 3.** *Write the proof that the greedy algorithm for submodular maximization produces a $(1 - 1/e)$ approximation.*

[Comment: The main elements of the proof are on the slides. You have to combine them into a formal proof.]

**Q 4.** *Suppose that we are doing social sensing. There are a set of social network accounts that publish news, which are reshared successively by others. Suppose we also have a probability on each edge that says how likely news is to propagate along that edge (i.e. shared by one node when it sees news shared by the other). How would you select your sensing nodes, or $k$ nodes to follow, so as to maximize expected coverage of news?*

**Problem instances** An *instance* of a problem is the problem asked for a particular dataset. For example, finding shortest path is an algorithmic problem, while finding the shortest path between a specific pari of nodes on a particular network is an instance of the shortest path problem.

**NP hardness.** (optional. not in exam). There are certain problems that are considered NP-hard, and belong to the class of problems called NP-hard. Let us refer to this set as $NPH$.

It can be shown that for any pair of problems $A, B \in NPH$, any given instance of $A$ can be "reduced" to an instance of $B$ in polynomial time. Meaning that given an instance of $A$, we can convert it to an instance of $B$ in polynomial time, and then any solution of the instance of $B$ can be converted back to a solution of the instance of $A$ in polynomial time.

This also implies that if there is a polynomial time solution to $B$, then that can be used to get a polynomial time solution of $A$ via the reduction. Thus, if any problem in $NPH$ has a polynomial time solution, then every problem in NPH will have a polynomial time solution via the reduction. It is however generally believed that NP-hard problems cannot have polynomial time solutions. Though a proof of this fact is not known.

When we encounter a new problem $C$, the usual way to show that it is NP-hard is to take a known problem $B \in NPH$ and show that a polynomial time reduction exists from $B$ to $C$. This implies that $C$ is NP hard, and a polynomial time algorithm is unlikely. If a poly time algorithm is found for $C$, that would imply that all problems in $NPH$ has poly time algorithms. Thus, once we have shown a problem to be NP-hard, finding ploy time algorithms for it is really unlikely.